GENERATING VALID SMILES SEQUENCES USING GENERATIVE DEEP MODELS

Pratheeksha Nair

Master of Technology Thesis

June 2019

International Institute of Information Technology, Bangalore

GENERATING VALID SMILES SEQUENCES USING GENERATIVE DEEP MODELS

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Technology

by

Pratheeksha Nair
IMT2014040

International Institute of Information Technology, Bangalore
June 2019

Dedicated to

all struggling souls striving to make a breakthrough in molecule generation and drug discovery

Thesis Certificate

This is to certify that the thesis titled **Generating valid SMILES sequences using generative deep models** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Pratheeksha Nair**, **IMT2014040**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Dinesh Babu

Bengaluru,
The 12th of June, 2019.

GENERATING VALID SMILES SEQUENCES USING GENERATIVE DEEP MODELS

Abstract

Drug discovery seeks to generate chemical species tailored to particular and very specific properties. Existing literature shows that the deep learning models used for this task exhibit scope for improvement in terms of the validity of generated drug molecules. Recurrent Neural Networks (RNN) and Variational Autoencoders (VAE) are among widely used architectures in this area largely due to the string-like representation of molecules called SMILES and due to their previously established success in sequence generation tasks. These SMILES strings follow a grammar with a certain set of syntactic and semantic rules and generating valid strings following these rules is not an easy task. This thesis proposes a novel training strategy, involving a validity factor, to improve the validity of SMILES sequences generated by these deep generative architectures. RNNs and VAEs were trained on 50,000 SMILES strings (from ZINC dataset) using the proposed strategy and improvements of 4.48% and 49% respectively were observed in the validity. Thus, this work shows that the proposed validity factor can be exploited in deep generative models to improve the quality of their results in terms of valid sequences generated.

Acknowledgements

This Master's Thesis would not have been completed successfully in the absence of a long list of people. Without the adequate computational resources, I would not have been able to run any experiments which were crucial for testing my hypothesis. I am grateful to IIIT-B and MINRO lab for providing the required computing power and a suitable workspace. I also express sincere gratitude to Vineet Reddy and Shubham Kumar for making these resources easily accessible. Timely advice and guidance by my supervisor, Prof. Dinesh Babu Jayagopi, helped ensure that I stuck to a timeline and plan of action, ultimately leading my work to the shape it is in today. I also thank my family for their patience and motivation throughout the process which helped keep a climate of calm. A special mention of thanks to Anup Deshmukh for constantly believing in the potential of my work and for engaging in productive brainstorming sessions. Lastly, I am grateful for being able to maintain my positiveness, rationality and perseverance throughout the duration of this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

AI . . . . . Artificial Intelligence

CFG . . . . . Context-Free Grammar

CVAE . . . . . Conditional Variational Autoencoder

ECFP . . . . . Extended Connectivity FingerPrint

GAN . . . . . Generative Adversarial Network

GRU . . . . . Gated Recurrent Unit

GVAE . . . . . Grammar Variational Autoencoder

HTS . . . . . High Throughput Screening

IIITB . . . . . International Institute of Information Technology Bangalore

InChI . . . . . International Chemical Identifier

LSTM . . . . . Long Short Term Memory

ML . . . . . Machine Learning

ORGAN . . . . . Objective Reinforced Generative Adversarial Network

RL . . . . . Reinforcement Learning

RNN . . . . . Recurrent Neural Network

S-DVAE . . . . . Syntax-Directed Variational Autoencoder

SMILES . . . . . Simplified Molecular-Input Line-Entry System

VAE . . . . . Variational Autoencoder

VI . . . . . Variational Inference

ZINC . . . . . ZINC Is Not Commercial

CHAPTER 1

INTRODUCTION

Although advancements in medicinal chemistry has particularly *changed the way you live and die* [1], there are still problems that are extremely difficult to solve and creating novel drugs is one of then [2]. The development of a new drug begins with scientists learning of some biological target (enzyme, protein, gene, etc) involved in a biological process whose mode of action differs from existing medicines. Identifying chemical compounds or molecules that are active towards these targets is quite challenging for chemists as it involves scouring the entire search space of nearly 10^{60} molecules [3]. The chosen molecules must be active in terms of molecular binding with biomolecules or elicit anticipated physiological responses on ingestion as drugs.

Current strategies to explore and discover molecules lie with high-throughput screening (HTS) [4] and genetic algorithms [5]. Although modern technology allows HTS at the scale of 10^4 [4], larger experiments turn out to be exceedingly expensive. HTS is the process by which a large number of compounds is automatically tested for some activity (activation, inhibition, etc) with the goal of identifying high-quality “leads" [6]. In order to achieve this, an enormous library of compounds is generated and then potential molecules are searched among the existing millions for closeness with known drugs using some similarity-based metrics [7]. The idea that bases this method is that in the molecular space, molecules closer to actual drugs may have stronger drug-like

properties. Thus, having computational tools that can narrow down the search space can prove very useful in overcoming practical limitations.

The task at hand aims at generating novel molecules (*de-novo* drug design) automatically. The end-to-end process involved in producing just one new drug manually takes upto 15 years and costs over 500 million dollars [8]. *Drug discovery* is the first stage in this process. In this stage, potential drugs are identified as candidates which then continue to clinical trials. The steps involved in this process can be seen below in FC1.1 [9].

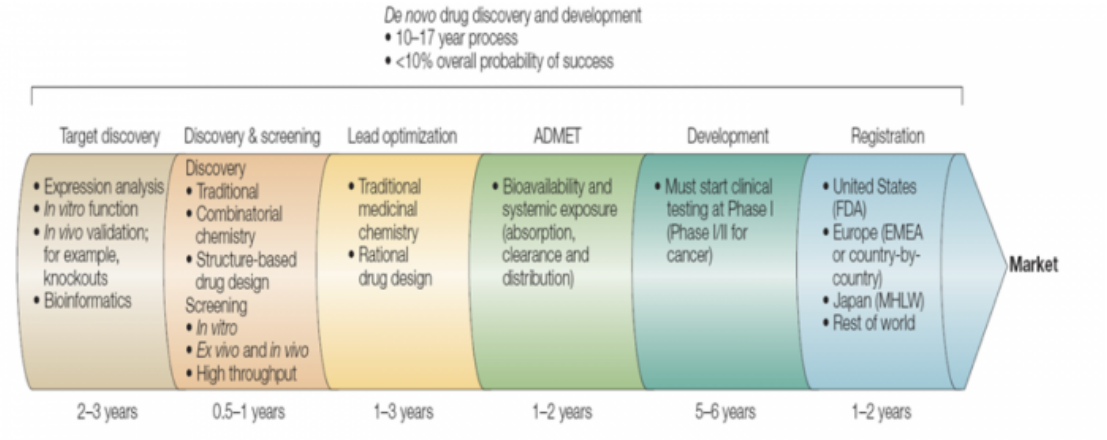

Figure FC1.1: The drug discovery pipeline

This problem of molecular discovery can also be posed as one of “inverse-design" [7] where fit candidates are generated specific to a set of properties and rules. Developments in Deep Learning and Artificial Intelligence (AI) such as Generative Adversarial Networks (GAN), sequence generation models and Reinforcement Learning (RL) have accelerated the feasibility of inverse-design. In this context, AI-based models are trained to generate new lead compounds such that their chemical and medical properties are predicted in advance.

There has been some research in the field of drug discovery using AI (as discussed in later chapters). Yet, there is a gap between the quality of results obtained and that

demanded by the community. To train generative models for this task, molecules are often represented as strings. This is largely due to the existence of powerful models for text sequence modelling [10]. However, these string representations of molecules are extremely brittle in the sense that small changes in the string can lead to completely different molecules. Moreover, these strings follow certain rules of syntax and semantics which may not be straightforward for a model to learn, thus leading to the generation of invalid strings. Such invalid strings cannot be decoded into plausible molecules.

Although there are works that address this specific problem like Grammar Variational Autoencoder (GVAE) [10] and Syntax-Directed Variational Autoencoder (S-DVAE) [11], out of the large number of molecules being generated by these models, the number of valid ones that are realizable in real-life, is small. Due to the complex nature of rules followed by the bonds and chains in chemical compounds, it is difficult for deep learning models to learn them inherently. Hence there is a need to explicitly supply the model with information regarding these rules. This idea was introduced by Dai et al. in their work S-DVAE [11].

Conceptually, structure generation requires formalization of the structure and the parameters of the stochastic processes involved are produced by a deep learning model [11]. These deep generative models, with the help of sufficient training examples, should be able to prefer valid sequences and shift the distribution towards the desired region in molecular space automatically. However, the problem of generating data that is correct in both syntax and semantics is still open.

A novel approach that hasn't been used for solving the problem of generating valid molecules is by training the model specifically for this. Existing research uses RNNs [12, 13], autoencoders [11, 14] and GANs [15] which do not explicitly penalize the model for generative invalid molecules. The RNN and autoencoder based methods are trained to reconstruct valid SMILES (Simplified Molecular-Input Line-Entry System)

strings while the GAN includes a discriminator that gives a higher reward for valid strings generated. However, the results obtained by these models indicate that there is scope for improvement. These are described in detail in the literature survey section (Chapter 2) and are compared with the proposed approach in the later sections. An important point to note is that these methods also address diversity of generated molecules, drug-likeliness, solubility and other specific properties. However, this thesis is focused solely on improving the validity of the generated strings, which is an independent problem in itself, and all comparisons carried out are on this basis.

The rest of this document is divided into six main chapters. Chapter 2 surveys the literature related to AI in drug discovery and describes the research work important to the development of this thesis. Chapter 3 discusses the various kinds of molecular representations popularly used for computation. In Chapter 4, the datasets used for this work and those commonly used in drug discovery, are examined. A brief overview of all the deep learning networks and tools used in this work follows in Chapter 5. The remaining Chapters 6 and 7 detail the experiments carried out with results and provide conclusive insights respectively.

CHAPTER 2

LITERATURE SURVEY

This chapter describes literature that has helped develop my hypothesis. This survey details the strategies and major claims of the works being studied. This is an important step in identifying the advantages, limitations and challenges introduced by these papers to solve the problem. A variety of deep learning based solutions have been employed to generate novel drug molecules such as Adversarial Autoencoders (AAE) [14], Variational Autoencoders (VAE) [11], sequence to sequence models (RNN) [12, 16], GAN [7] and RL [17, 18]. This section will deal in detail with three of the above mentioned architectures, namely, RNNs, VAEs and GANs.

The work on AAE [14] is specific to generating anti-cancer drugs. It makes use of a 7-layer autoencoder (encoder-decoder) setting where the middle layer acts as a discriminator. The discriminator is trained to differentiate between a given latent distribution and one from the encoder. Meanwhile, the encoder is trained to generate representations that can confuse the discriminator. In this manner, it follows an adversarial training while the encoder and decoder are also trained jointly to work as an autoencoder. The latent layer also contains a neuron known as a “growth inhibitor". A negative value of the growth inhibitor means that the number of tumour cells have reduced after treatment with a particular drug. The molecules are represented by fixed length binary vectors known as binary fingerprints. Since this method is very specific to anti-cancer

drugs, it does not fall directly under our interest. Hence the proposed approach is not deployed on this model and the results obtained are not compared.

RL based techniques were used in conjunction with RNNs by Jaques et al. [17] to generate SMILES with molecular properties like solubility, etc in a desirable range. However, the reward function is dependent on manually written rules for penalizing undesirable structures. To overcome this, Olivecrona et al. [18] introduced a policy-based RL algorithm to fine tune pre-trained RNN models for molecular generating.

The following sections discuss in detail, the works by Segler et al. using RNNs [12], Dai et al. [11] involving Syntax-Directed Variational Autoencoder (S-DVAE) and Sanchez et al. on Objective Reinforced Generative Adversarial Network (ORGAN) [7]. The results from the proposed work in this thesis are compared primarily with those obtained by using these architectures.

2.1 Generating Focussed Molecule Libraries for Drug Discovery with RNNs

This work by Segler et al. [12] shows that RNNs can be trained as generative models for molecular structures. Generative models learn the probability distribution over the training data and sampling from this distribution generates new results close to the training data. This work is based on the hypothesis that a generative molecule trained on drug molecules will “know" how valid drugs look like and thus generate them.

Long Short Term Memory (LSTM) based RNNs are trained on a corpus of drug molecules represented as SMILES strings (more details in Chapter 3). SMILES is a formal grammar that describes molecules with ah alphabet of characters, numbers representing rings and brackets indicating chains. To generate valid SMILES, the RNN will need to learn this grammar. This work encodes the input SMILES as one-hot rep-

resentations [19]. That is, if there are K symbols, and at time step t, k is the input symbol, then the input vector x_t can be constructed with length K with only the k-th position as one and all other entries zero.

The probability distribution $P_\theta(s_{t+1}|s_t,...,s_1)$ of the next symbol given the previous sequence is estimated using the output vector y_t of the RNN at time step t by

$$P_\theta(s_{t+1}|s_t,...,s_1) = \frac{\exp y_t^k}{\sum_{k'=1}^{K} \exp y_t^{k'}} \quad \text{(Eqn 2.1)}$$

where y_t^k is the k-th element of vector y_t. Sampling from this distribution generates novel molecules.

Furthermore, their work also demonstrates that RNNs can transfer the knowledge learned by training on large molecule sets to produce novel molecules by retraining on small sets of known actives. They test their model on a corpus of active molecules and report close to 97% validity of SMILES strings generated.

2.2 S-DVAE for Structured Data

Conceptually, generation of structured data includes formalization of the structure and a deep generative model producing parameters for stochastic process [11]. This is based on the belief that with sufficient training examples, the loss function will prefer valid patterns. Conditional Variational Autoencoder (CVAE) [16] applies sequence models to convert complex structures into sequences for chemical molecule generation, using SMILES line notation. However, the lack of formalized syntax and semantics serves as a restriction on the structure generation. In Grammar Variational Autoencoder (GVAE) [10], Kusner et al. incorporate structure restriction explicitly by taking into account the Context-Free Grammar (CFG) of SMILES. Althoug this handles syntactic

validity, is it still incapable of regularizing the model for semantically valid generation. S-DVAE [11] tackles this challenge by introducing *stochastic lazy attributes* that enables converting offline semantic check to online.

A CFG is defined as $G = \langle V, \Sigma, R, s \rangle$ where V is the set of non-terminal symbols, Σ is the set of terminal symbols, R is the set of production rules and $s \in V$ is the start symbol. Each production rule determines transition from a non-terminal symbol to a sequence of terminal and/or non-terminal symbols.

Attributes attached to non-terminal symbols in CFGs formalized by Knuth et al. [20] enriches it with “semantics". These attributes may be *inherited* from its parents and siblings or *synthesized* from the attributes of its children. This formalism is known as attribute grammar [20].

Production	**Semantic Rules**
$\langle s \rangle \rightarrow \langle atom \rangle_1 \text{`}C' \langle atom \rangle_2$	$\langle s \rangle.matched \leftarrow \langle atom \rangle_1.set \cap \langle atom \rangle_2.set$
$\langle atom \rangle \rightarrow \text{`}C' \mid \text{`}C' \langle bond \rangle \langle digit \rangle$	$\langle s \rangle.ok \leftarrow \langle atom \rangle_1.set = \langle s \rangle.matched = \langle atom \rangle_2.set$
$\langle bond \rangle \rightarrow \text{`}-' \mid \text{`}=' \mid \text{`}\#'$	$\langle atom \rangle.set \leftarrow \phi \mid concat(\langle bond \rangle.val, \langle digit \rangle.val)$
$\langle digit \rangle \rightarrow \text{`}1' \mid \text{`}2' \mid ... \mid \text{`}9'$	$\langle digit \rangle.val \leftarrow \text{`}1' \mid \text{`}2' \mid .. \text{`}9'$

The production rules of the CFG are shown above left and the semantic rules responsible for calculating the attribute values are to the right. These have been highlighted in Dai et al.’s work on S-DVAE [11]. The attribute grammar is used to check the semantic rules such as opening and closing of rings, etc. Such dependencies are not context-free [11].

For a given SMILES string, its parse tree may be generated from the CFG and for each node in the tree, the corresponding attribute values may be checked and the overall validity of the string may be determined. However, while generating a string, there is no tree to begin with. For example, when extending the start symbol $\langle s \rangle$, none of its

children are generated and hence $\langle s \rangle$.matched cannot be computed. To address this issue, *stochastic lazy attributes* were introduced. Whenever there is lack of a synthesized attribute due to absence of children nodes, a stochastic attribute (a value sampled from a Bernoulli distribution) is given to the node under consideration. These stochastic attributes are then passed on to its children nodes to ensure semantic validation during tree generation. Now, a non-terminal node v whose semantic rules are either satisfied or has stochastic attributes is chosen and then a rule from the production rules of the CFG is sampled according to the distribution $p_\theta(r|v,T)$ where r is the rule and T is the syntax tree built so far. Based on the production rules, the non-terminal can be expanded and the symbol can be added as children to node v.

In this manner, the generation process of the SMILES string follows the semantic constraints and the syntactic rules (by following production rules of CFG) at every step. The architecture employed is that of a variational autoencoder where the encoder approximates the posterior of the latent variable and the tree generation procedure samples from the decoder probability. The paper reports that 43.5% of generated SMILES were valid.

2.3 ORGAN for Sequence Generation Models

This paper [7] introduces a novel approach to optimize the properties of a distribution of sequences. The ORGAN includes a generator that maximizes a weighted average of the objective function and domain-specific metric. These two types of rewards are considered together. The discriminator is trained along with the generator in an adversarial fashion [21]. This idea is implemented on top of the SeqGAN [22] architecture that combines GANs and RL for sequential data generation.

In SeqGan, the generator is trained to produce sequences and the discriminator is trained to distinguish between real and generated sequences. However, the sampling

process is not differentiable for discrete data. The generator is thus trained as an RL agent whose reward function $R(Y_{1:T})$ is defined for full-length sequences. Given an incomplete sequence $Y_{1:t}$, G_θ must produce an action a, along with the next token y_{t+1}. The agent's stochastic policy is given by, $G_\theta(y_t|Y_{1:t-1})$ and its expected long term reward is to be maximized. The long term reward is represented by the following equation:

$$J(\theta) = E[R(Y_{1:T})|s_0, \theta] = \sum_{y_1 \in Y} G_\theta(y_1|s_0).Q(s_0, y_1) \quad \text{(Eqn 2.2)}$$

where s_0 is a fixed initial state, $Q(s,a)$ is the action-value function representing the expected reward at state s of taking an action a and following the current policy G_θ to complete the rest of the sequence.

In order to account for the domain-specific desired objective, the reward function for a particular sequence $Y_{1:t}$ is extended to a linear combination of D_ϕ and O_i, parametrized by λ as follows:

$$R(Y_{1:T}) = \lambda.D_\phi(Y_{1:T}) + (1-\lambda).O_i(Y_{1:T}) \quad \text{(Eqn 2.3)}$$

If $\lambda = 0$, the model ignores D and becomes a "naive" RL algorithm, whereas if $\lambda = 1$, it is simply a SeqGAN model. Here, O_i corresponds to domain-specific objective values. For molecules, some of the objectives chosen were solubility (in water), synthesizability (how hard is it to synthesize the given molecule) and druglikeliness (having drug-like properties).

All the above discussed methods, although demonstrating promising results, leave scope for improvement in terms of validity of generated strings. The hypothesis of including a validity score in training these models is formulated and tested. This modified training strategy presents increase in the validity. Details are described in Chapter 6.

CHAPTER 3

MOLECULAR REPRESENTATIONS FOR COMPUTATION

In recent times, automatic chemical property predictions using machine learning has become popular in the drug discovery community [23–26]. Majority of these algorithms take fixed-length vectors as input [27,28] although this is a difficult task [29,30]. Molecules differ in types of atoms, bonds, etc, which makes it tricky to represent them using fixed-length vectors. Overall, the choice of the representation of molecules is imperative for machine learning-based drug discovery [31].

Traditionally, fixed-length vector representations of molecules, named fingerprints, are designed based on human expertise knowledge and is not data-driven [25, 32]. One example is based on hashing procedures known as Extended Connectivity FingerPrint (ECFP) [33]. Such fingerprints are obtained by a lossy compression [34] and are efficient in speed and non-invertible. Another non-data-driven fingerprint is based on local sub-structures of molecules. Fingerprint feature vectors are designed for specific tasks using highly related molecular sub-structures [35, 36]. However, this kind of design is highly task-dependent and requires years of expertise and experience. Thus, non-data driven fingerprints are either unable to encode enough information or are highly dependent on human knowledge.

Chemical structures may be represented by linear notations which encode their stereochemisty as text [37]. These can be also leveraged to store and check the iden-

tity of molecules. Such representations are compact, human-readable and can be easily entered into software (like text entry boxes in websites, spreadsheet cells, etc.). Each molecule may have a unique representation (canonical), thus making it easier to identify them and search for them in the web and chemical databases. [38].

Two of the most widely used line notations are Weininger's SMILES string [39] also developed by Daylight Chemical Information Systems [40], and InChI (International Chemistry Identifier) representation from IUPAC [41].

3.1 SMILE Representation of Molecules

The SMILES format is the most popularly used line notation. In this system, the atoms, bonds and rings of chemical structures are encoded in a graph [31]. For example, dinitrogen and methyl isocyanate are represented as N$\equiv$N (N#N) and CH3N$=$C$=$O (CN=C=O) with corresponding SMILE representations in brackets. The Figure FC3.1 [31] shows some more examples of the SMILE representation.

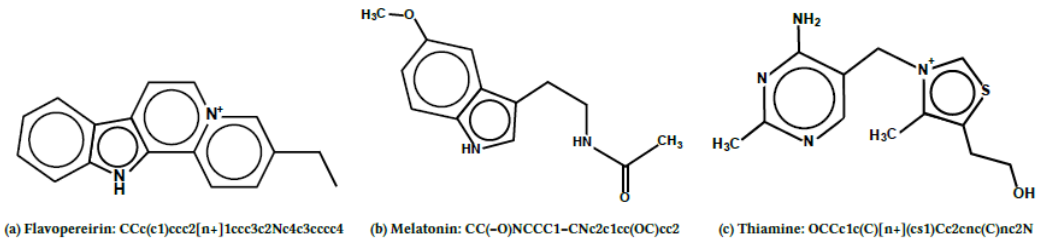

Figure FC3.1: Examples of SMILE representations

However, there are some drawbacks of the SMILES format as highlighted by Noel M O'Boyle [38]. It cannot be used to represent molecules with more than 2 valence electrons. It also fails to support a variety of stereochemistry types. Also, aromaticity cannot be handled in a standard way. Moreover, canonical representations cannot be generated. A canonical representation is unique in the sense that each structure is always

represented by the same line notation [42].

3.2 InChI Representation of Molecules

There was a need for a standard linear canonical representation which led to InChI (proposed by Steve Heller and Steve Stein at the National Institute of Standards and Technology in the US) [38]. The main design modification was in allowing information linking of the same molecule across various databases and generating a canonical representation. To do this, the InChI algorithm identifies isomers using a layered network. Isomers are chemical compounds with the same formula but different spatial arrangement of atoms within the molecule, often leading to different properties.

3.3 Fingerprint based methods

Hash-based fingerprints ECFP [33] is one of the most popular hash-based method. Circular fingerprints generate features from each iteration by applying a fixed hash function on the features of the previous iteration. However, they do not capture enough information as oftentimes the hash function can not be inverted. This results in lower performance in any predictive task [31].

Biologist-guided Local-Feature Fingerprints Biologist identify task-related sub-structures and produce fingerprints by using their count as local features [35]. This kind of fingerprint methods is usually highly task-specific and are not generalizable [31].

Supervised Deep Learning-based Fingerprints Molecular fingerprint are learned from data samples using deep learning techniques, without explicit human guidance [43–45]. The current best is the neural fingerprint [44] which is obtained by a process similar to

that of generating circular fingerprints. Here, a fully connected year with a non-linear activation takes the place of the hash function. However, this requires large amounts of data and in order to acquire this, a huge number of expensive tests need to be carried out on the molecules [31].

Seq2seq fingerprint Xu et al. [31] introduce a deep-learned based representation called seq2seq fingerprint. This format is obtained by training a Gated Recurrent Unit (GRU) based sequence-to-sequence model. This network maps the input molecular string to a fixed-size vector which is then fed to another deep GRU network that generates the original string from the seq2seq fingerprint.

These fingerprints are obtained by additional processing on SMILE strings. SMILE, although not without drawbacks, is one of the most commonly used and easily readable representations of molecules. They are essentially strings which can be recognized directly by powerful sequence-to-sequence models allowing for strong analysis and further processing. The major advantages of SMILES format include how easily it can be read, learnt and understood by humans [38]. However, a lack of published formal specification has caused several ambiguities leading to differences in implementation. Moreover, there also exist large amount of data in the SMILE format which can be leveraged for deep learning tasks. More information on the datasets used commonly can be obtained in the following Chapter 4.

CHAPTER 4

DATASET

In order to fully understand all levels of complex diseases, multiple laboratories and institutes need to collect, assess and understand the required data, such that statistical conclusions may be reached [46]. Discovering new drugs and interpreting diseases are subject to unifying analysis of data from various sources.

For the purpose of this thesis, a corpus of valid molecular strings was required in the SMILE format. Different deep learning architectures were to be trained on a large number of valid SMILE string samples in order to generate new valid strings. Experiments were run using the ZINC (recursive acronym - ZINC is not commercial) database [47].

ZINC is a continuously growing library of 727,842 molecules, and their 3D structure, obtained from catalogs of compounds from vendors [47]. This dataset was created with the intention of making access to purchasable chemical compounds easy, in order to facilitate virtual-screening. The database also includes properties such as molecular weight, calculated LogP, and number of rotatable bonds of the molecules. This dataset is available for free download and is supported by a web-based tool used for querying, searching, browsing through the database and creating subsets. It contains many “drug-like" and “lead-like" molecules which are useful for tasks like drug discovery.

The creators of the ZINC database shed some light on smaller details of data collection and filtering [47]. 10 vendor catalogs were used. The dataset was cleared of

molecules whose formula weight is greater than 700, whose number of hydrogen-bond donors are greater than 6 and number of hydrogen bond acceptors greater are than 11. Molecules with solubility measure (LogP) in the range of -4 and 6 were only included while the others were removed and the number of rotatable bonds of all molecules were lesser than 15. $H, C, N, O, F, S, P, Cl, Br, I$ was the set of allowed atoms within the molecules. The only inclusions that violated these constraints were actual drugs. These rules allow the database to roughly comply with expert opinions regarding “drug-like" properties. The molecules obtained were also converted to canonical forms using tools of OpenEye Scientific Software (http://www.eyesopen.com). This ensures unique representation for a particular molecule.

To run the experiments included in this work, a subset of SMILE strings from ZINC dataset were used. The details of the data used are included in Table TC4.1.

Table TC4.1: Details of ZINC dataset used

Training set size	50,000
Num. of valid points	50,000
Max. data length	162
Size of alphabets	34

CHAPTER 5

DEEP LEARNING MODELS

In order to test the hypothesis, a variety of deep learning models have been used. This chapter deals with briefly describing these models and their working. It is important to understand the intuition and functioning of these models for a better overall perception of the task at hand and a better interpretation of the results. The following models are being discussed.

- RNN
- LSTM
- AE
- VAE
- GAN

5.1 RNN

RNNs are dynamic models used to generate sequences. They can be trained to predict what comes next given real data sequences from each of the previous time steps. The trained networks can generate novel sequences by sampling from their output distribution and sending these sample as inputs in the following time step.

Typically, RNNs are used in cases where temporal data is to be modelled because of their inherent internal memory [19]. Loosely speaking, they remember important information from both the input they received previously and currently in order to give very precise predictions of the future. An RNN takes the present and the past as its inputs. Sequential data contains crucial information about what comes next and this is leveraged by RNNs. Weights are applied to both current input as well as previous inputs and these weights are updated through gradient descent and backpropogation through time. RNNs may be considered as a series of Neural Networks trained successively with backpropogation.

An input vector sequence $\mathbf{x} = (x_1, x_2, ...x_T))$ goes through multiple hidden layers that are recurrently connected to compute the hidden states $\mathbf{h}^n = h_1^n, h_2^n, ...h_T^n$ and the output vector $\mathbf{y} = (y_1, y_2, ...y_T)$. Each output vector y_t is used to obtain a distribution $Pr(x_{t+1}|y_t)$ to sample the next possible input x_{t+1}.

The following equations (Eqn 5.1 and Eqn 5.2) are used to calculate the hidden layer activations. t iterates from 1 to T and n from 2 to N.

$$h_t^1 = \mathscr{H}(W_{ih^1}x_t + W_{h^1h^1}h_{t1}^1 + b_h^1) \quad \text{(Eqn 5.1)}$$

$$h_t^n = \mathscr{H}(W_{ih^n}x_t + W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t1}^n + b_h^n) \quad \text{(Eqn 5.2)}$$

Here the W terms are weight matrices, the b terms indicate bias vectors and $\mathscr{H}$ is the function of the hidden layer.

Given the hidden states, the output $\hat{y}_t$ is computed as follow,

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^ny}h_t^n \quad \text{(Eqn 5.3)}$$

$$y_t = Y(\hat{y}_t) \quad \text{(Eqn 5.4)}$$

where Y is the output layer function.

The probability of the input sequence $\mathbf{x}$ is

$$Pr(\mathbf{x}) = \Pi_{t-1}^{T} Pr(x_{t+1}|y_t) \quad \text{(Eqn 5.5)}$$

and the sequence loss $\mathscr{L}(\mathbf{x})$ is:

$$\mathscr{L} = -\sum_{t=1}^{T} logPr(x_{t+1}|y_t) \quad \text{(Eqn 5.6)}$$

5.2 LSTM

There are two major obstacles RNN's deal with – exploding gradients and vanishing gradients. The former happens when the weights are assigned high importance and their updates keep getting multiplied leading to very large values. The latter happens when the value of the gradients are too small which stops the model from learning or it converges too slow. Also, in practice, RNNs are unable to store information about past inputs for very long. Having a longer memory has a more stabilising effect. LSTM [48] is an RNN architecture shown to address the problems of exploding and vanishing gradients.

LSTM architecture uses cells with memory for storing information and for finding long range dependencies in the data. Using vanilla LSTMs, $\mathscr{H}$ is implemented by the following functions.

$$
\begin{aligned}
i_t &= \sigma(W_{x^i}x_t + W_{h^i}h_{t1} + W_{c^i}c_{t1} + b_i) \\
f_t &= \sigma(W_{x^f}x_t + W_{h^f}h_{t1} + W_{c^f}c_{t1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{x^c}x_t + W_{h^c}h_{t1} + W_{h^c}c_{t1} + b_f) \\
o_t &= \sigma(W_{x^o}x_t + W_{h^o}h_{t1} + W_{c^o}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
\qquad \text{(Eqn 5.7)}
$$

here σ in Eqn 5.7 is the logistic sigmoid function, and the input gate, forget gate, output gate, cell and cell input activation vectors, all of which are the same size as the hidden vector h are represented by i, f, o and c respectively .

5.3 AE

Autoencoders are artificial neural networks used to learn efficient data codings in an unsupervised manner [49]. The representation also called encoding learned is typically of reduced dimension. Along with the dimensionality reduction, a network is also included which learns reconstruction of the input.

An AE tries to learn a function $h_{W,b}(x) \approx x$. It approximates an identity function, so as to output $\hat{x}$ that is similar to x. It compresses the input to a latent-space representation and then reconstructs the output from this representation. It consists of an encoder and a decoder. Encoder is the part that compresses the input into a latent-space representation. The decoder is the part that reconstructs the input from the latent space representation. The idea behind AE is that the latent representation h can help identify and analyse meaningful properties. AEs are trained to minimise the reconstruction error as given by

Eqn 5.8.

$$\mathscr{L}(x,x') = ||x - x'||^2 = ||x - \sigma'(W'(\sigma(Wx+b)) + b')||^2 \quad \text{(Eqn 5.8)}$$

W is a weight matrix and b is a bias vector for the encoder whereas W' and b' are the corresponding parameters for the decoder.

5.4 VAE

A VAE model is very similar to the AE where an encoder encodes the input to a vector or scalar, then a decoder decodes the vector back to its original form [31]. The difference lies in the assumption made by the the VAE model that the embedded space follows a Gaussian distribution. If $P(X)$ is the true distribution of the data, $P(z)$ is the distribution of latent variable where z is the latent variable, and $P(X|z)$ is the distribution of generating data given latent variable, then $P(X)$ can be given by Eqn 5.9.

$$P(X) = \int P(X|z)P(z)dz \quad \text{(Eqn 5.9)}$$

The idea of VAE is to infer $P(z)$ using $P(z|X)$. $P(z|X)$ using a popular choice of method in bayesian inference called Variational Inference (VI). VI proposes to pose the inference problem as an optimization problem. One approach to this is by modeling the true distribution $P(z|X)$ using a known distribution that is easy to evaluate, like Gaussian, and minimizing the difference between those two distribution using KL divergence metric (which measures the distance two distributions are) [11]. The final VAE objective function is given by,

$$log(P(X)) - D_{KL}[Q(z|X)||P(z|X)] = E[log(P(X|z)] - D_{KL}[Q(z|X)||P(z)] \quad \text{(Eqn 5.10)}$$

where $Q(z|X)$ is the known Gaussian distribution.

5.5 GAN

GANs [21] consist of a generator G and a discriminator D which are trained via an adversarial process. G captures the distribution of the data and D estimates the probability that a sample is true or false. If a sample is drawn from the training data, then it is a true sample and if it is generated by G, then it is a false sample. The training strategy for G is to maximize the probability of fooling D and making it wrongly classify a sample as true or false. Thus, they play a two-player minimax game. Here, both G and D are represented by multilayer perceptrons and the entire system can be trained with backpropagation.

To learn the generator's distribution p_g over data $\mathbf{x}$, the input noise variables $p_z(z)$ is assumed to follow a prior distribution. A generator $G(z;\theta_g)$ function is then used to map the prior to the data space and is parameterized by θ_g. Another function $D(x;\theta_d)$ that outputs whether its input is true or false, is also defined. It approximates the probability that the input $\mathbf{x}$ was sampled from the data or generated by G. D is trained to maximize the probability of correctly classifying its input samples. G is trained to minimize $log(1-D(G(z)))$. Thus, D and G try to optimize the following value function $V(G,D)$,

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[logD(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1-D(G(\mathbf{z})))] \quad \text{(Eqn 5.11)}$$

CHAPTER 6

PROPOSED APPROACH, EXPERIMENTS AND RESULTS

This chapter discusses the main contributions of this thesis, the experiments run to test the hypothesis and the results obtained.

6.1 Hypothesis

The hypothesis of this work is that *the validity of SMILE strings generated using deep generative models may be improved by explicitly training these models for validity with the help of a validity factor*.

Generally speaking, the deep generative models used for this task are trained to minimize a reconstruction loss (using RNNs and VAEs) or a minimax loss (using GANs). Naturally, a model that is being trained to reconstruct valid SMILE strings (fed during training) learns a latent molecular space from which sampling during inference, results in new and valid strings. However, what if during training, the model was explicitly told when it was decoding valid strings and when it wasn't? This could enable the model to focus on not only reconstruction but also valid string generation. One way to achieve this is by explicitly penalizing the model when it generates an invalid string and rewarding it for every valid string produced. The simplest way of doing this is by modifying the loss function.

In this work, a *validity factor* vf is introduced which parameterises a function $\mathscr{F}$ that modifies the loss value accordingly. The validity factor at each training step determines whether the model will be penalized or rewarded. The modified loss function for iteration i will be obtained by the following equation,

$$L_{new_i} = \mathscr{F}(L_i, vf_i) \tag{Eqn 6.1}$$

where L_i is the reconstruction loss value for the ith step. In the current setting, vf is designed to upscale or downscale the value of the loss L to obtain L_{new}. At each training step of the deep generative model, it is made to generate a string which is then checked for validity. If the generated string is invalid, then vf for that step is set to a value ≥ 1 and if it is valid, the value is set to ≤ 1. Ideally, value of vf needs to be set in such a way that the value of L_{new} follows the following matrix.

Table TC6.1: Table for loss values

Loss function values	**Invalid Sequence**	**Valid Sequence**
High value of L_i	Very high value of L_{new_i}	High value of L_{new_i}
Low value of L_i	High value of L_{new_i}	Very high value of L_{new_i}

In the case of the RNN [12], valid SMILE strings were fed as input, in order to reconstruct them. Thus the model minimizes the *reconstruction loss* between the generated sequence and the actual sequence. In VAE [11], there are both *reconstruction loss* and *KL-divergence* which are optimized by the model. In the new setting, these models will optimize the modified loss which will be greater in case of invalid string generation and lesser if the generated string is valid. Figure FC6.1 shows the block diagram for the proposed approach.

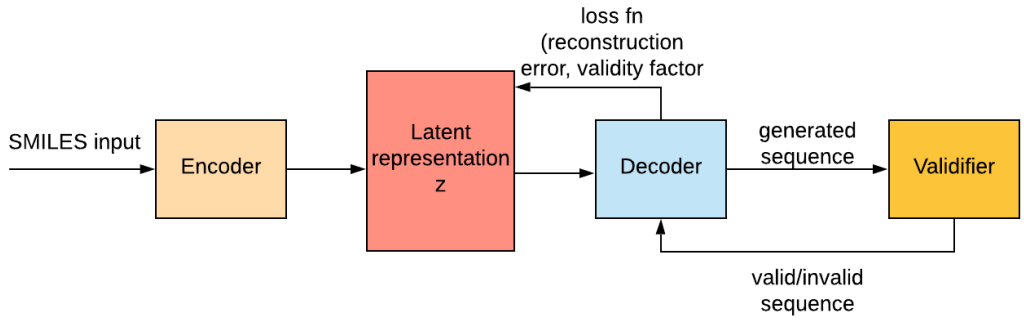

Figure FC6.1: Block diagram for proposed approach

6.2 Rules for validity

A generated SMILE string may be deemed valid if it satisfies a set of rules (both syntactic and semantic).

Atoms

1. Atoms are represented by their chemical formulas in square brackets
2. Brackets may be excluded if atoms
   (a) are B, C, N, O, P, S or a halogen (F, Cl, Br, I), and
   (b) do not have a formal charge
   (c) are regular isotopes

Bonds A bond is represented using:

1. . - = # $: / \
2. '.', to indicate that two parts are not bonded together ("non-bond")

Rings

1. Ring structures are indicated by numerical labels to show closing of rings
2. Multiple ring closings are shown by multiple digits
3. A bond type indicating the type of the ring-closing bond may be written before the digits
4. Multiple bonds may not be represented by ring-closing bonds

Branching

1. Branches are described with parentheses
2. The atom at the branch is attached to both the first atom inside the parentheses as well as the first atom after the parenthesis
3. Branches may be written in any order

Although this is not an exhaustive list of rules to be followed, it is essential that all valid SMILE strings follow this set of rules. For all experiments, the validity of generated strings is tested with the help of RDKit library[1] that inherently checks for all the rules. RDKit is an open source tool for chemoinformatics that supports APIs for Python, C++ and Java and has functionalities for almost all kind of operations involving molecules.

6.3 Experiments and Results

In order to test the hypothesis in section 6.1, the modified training strategy involving a validity factor was experimented upon two architectures, namely RNN [12] and VAE [11]. For all the experiments, the value of validity factor vf was taken as 2 and

[1] https://github.com/rdkit/rdkit

$\mathscr{F}(a,b) = a \times b$. Although these allocations do not strictly conform with Table TC6.1, they were chosen solely for the purpose of testing the hypothesis. Both models were trained on a dataset sample of 50000 SMILES (all valid) and while testing for validity, 200 new samples were generated.

6.3.1 Experiments with RNN

The RNN model implementation[2] used was based on the original paper [12]. The RNN model had three stacked LSTM layers (as done in the paper). The model was trained till convergence using a batch size of 128 and embedding dimension 248. The learning rate used was 0.005. To generate novel molecules, 200 sequences were sampled from the model and the percentage of valid SMILES was calculated. This sampling was repeated 5 times and the results shown in Table TC6.2 is an average value. The validity percentage has improved by 4.5%.

Table TC6.2: Validity results (%) using RNN with validity factor

Method	No. of samples	Validity (%)
RNN by Segler et al.	200	74.50
RNN with validity factor	200	**89.50**

6.3.2 Experiments with S-DVAE

The original S-DVAE implementation[3] mentioned in the paper [11] has been used for this experiment. The model was trained using a batch size of 300 for 500 epochs with a learning rate of 0.0001. 200 sequences were sampled 5 times and the average validity percentage obtained is given in Table TC6.3. There is an improvement of 49%.

[2]https://github.com/nair-p/Generate-novel-molecules-with-LSTM
[3]https://github.com/Hanjun-Dai/sdvae

Table TC6.3: Validity results (%) using S-DVAE with validity factor

Method	No. of samples	Validity (%)
S-DVAE by Dai et al.	200	27.50
S-DVAE with validity factor	200	**41.00**

6.4 Discussion

The experimental results show improvement in validity due to the modified training strategy involving a validity factor. Intuitively, this makes sense as the model is being penalized for every invalid sequence it generates at each step of the training and rewarded for generating a valid string. Since the loss values increase and decrease drastically, the model learns the valid molecular space representation faster and better.

The results obtained by the experiments are only preliminary, yet are promising. However, there are some points to be taken note of. The implications of the chosen validity factor and function $\mathscr{F}$ on the results, is to be studied. Other validity factors and functions need to be explored, keeping in mind the conditions of Table TC6.1. Although both RNN and S-DVAE are considered as the state-of-the-art methods for molecule generation, it would be interesting to test this training strategy on a GAN architecture like ORGAN [15]. Moreover, since this work is focused entirely on the validity of generated SMILES, other factors such as reconstruction and diversity of generated list of molecules, have not been considered. It is possible that such a strategy may negatively affect diversity.

Although this entire work was modelled around generating valid molecules, it can also be formulated in a more general manner, for producing arbitrary sequences that follow certain rules of syntax and semantics. This is an open problem and hence investigating the effects of the proposed training strategy on problems like program generation or any kind of structural sequence generation would bring forth intriguing insights.

CHAPTER 7

CONCLUSION

This thesis tested the hypothesis of explicitly training deep generative models for validity of SMILES. A modified training approach that involves a validity factor was introduced. According to this approach, at each step of training, for every invalid string generated, the model is penalized by increasing the loss value and for every valid string generated, the model is rewarded by reducing the loss value. A generated SMILE string is checked for validity against the set of pre-defined generative syntactic and semantic rules. This strategy was tested on two deep generative architectures – RNNs and VAEs. The experiments resulted in a 4.5% and 49.0% improvement in validity respectively. However, other factors like diversity and reconstruction are not considered. Also, the strategy needs to be examined more with multiple kinds of validity factors. Having mentioned these caveats, it should be noted that on a basic level, the hypothesis has been tested and validated.

Bibliography

[1] George M Whitesides. Reinventing chemistry. *Angewandte Chemie International Edition*, 54(11):3196–3209, 2015.

[2] Petra Schneider and Gisbert Schneider. De novo design at the edge of chaos: Miniperspective. *Journal of medicinal chemistry*, 59(9):4077–4086, 2016.

[3] Jean-Louis Reymond, Lars Ruddigkeit, Lorenz Blum, and Ruud van Deursen. The enumeration of chemical space. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(5):717–733, 2012.

[4] Gisbert Schneider and Karl-Heinz Baringhaus. *Molecular design: concepts and applications*. John Wiley & Sons, 2008.

[5] D Balamurugan, Weitao Yang, and David N Beratan. Exploring chemical space with discrete, gradient, and hybrid optimization methods. *The Journal of chemical physics*, 129(17):174105, 2008.

[6] James R Broach, Jeremy Thorner, et al. High-throughput screening for drug discovery. *Nature*, 384(6604):14–16, 1996.

[7] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alán Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *Harvard University, Chem Rxiv*, 2017.

[8] Shahar Harel and Kira Radinsky. Accelerating prototype-based drug discovery using conditional diversity networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 331–339. ACM, 2018.

[9] Ted T Ashburn and Karl B Thor. Drug repositioning: identifying and developing new uses for existing drugs. *Nature reviews Drug discovery*, 3(8):673, 2004.

[10] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954. JMLR. org, 2017.

[11] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

[12] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2017.

[13] Anvita Gupta, Alex T Müller, Berend JH Huisman, Jens A Fuchs, Petra Schneider, and Gisbert Schneider. Generative recurrent networks for de novo drug design. *Molecular informatics*, 37(1-2):1700111, 2018.

[14] Artur Kadurin, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, 8(7):10883, 2017.

[15] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[16] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[17] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1645–1654. JMLR. org, 2017.

[18] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

[19] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[20] Donald E Knuth. Semantics of context-free languages. *Mathematical systems theory*, 2(2):127–145, 1968.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[22] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[23] Mark Ashton, John Barnard, Florence Casset, Michael Charlton, Geoffrey Downs, Dominique Gorse, John Holliday, Roger Lahana, and Peter Willett. Identification

of diverse database subsets using property-based and fragment-based molecular descriptions. *Quantitative Structure-Activity Relationships*, 21(6):598–604, 2002.

[24] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

[25] Frank R Burden. Molecular identification number for substructure searches. *Journal of Chemical Information and Computer Sciences*, 29(3):225–227, 1989.

[26] Xiao Qing Lewell, Duncan B Judd, Stephen P Watson, and Michael M Hann. Recap retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. *Journal of chemical information and computer sciences*, 38(3):511–522, 1998.

[27] Feiyun Zhu, Bin Fan, Xinliang Zhu, Ying Wang, Shiming Xiang, and Chunhong Pan. 10,000+ times accelerated robust subset selection. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[28] Feiyun Zhu, Ying Wang, Bin Fan, Shiming Xiang, Geofeng Meng, and Chunhong Pan. Spectral unmixing via data-guided sparsity. *IEEE Transactions on Image Processing*, 23(12):5412–5427, 2014.

[29] Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem: Chemistry Enabling Drug Discovery*, 3(10):1503–1507, 2008.

[30] Thomas A Halgren. Merck molecular force field. iii. molecular geometries and vibrational frequencies for mmff94. *Journal of computational chemistry*, 17(5-6):553–586, 1996.

[31] Zheng Xu, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery. In *Proceedings of*

the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, pages 285–294. ACM, 2017.

[32] Ramaswamy Nilakantan, Norman Bauman, J Scott Dixon, and R Venkataraghavan. Topological torsion: a new molecular descriptor for sar applications. comparison with other descriptors. *Journal of Chemical Information and Computer Sciences*, 27(2):82–85, 1987.

[33] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[34] Zhongxing Peng, Zheng Xu, and Junzhou Huang. Rspirit: Robust self-consistent parallel imaging reconstruction based on generalized lasso. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 318–321. IEEE, 2016.

[35] Noel M OâĂŹBoyle, Casey M Campbell, and Geoffrey R Hutchison. Computational design and selection of optimal organic photovoltaic materials. *The Journal of Physical Chemistry C*, 115(32):16200–16210, 2011.

[36] Chetan Rupakheti, Aaron Virshup, Weitao Yang, and David N Beratan. Strategy to discover diverse optimal molecules in the small molecule universe. *Journal of chemical information and modeling*, 55(3):529–537, 2015.

[37] Wendy A Warr. Representation of chemical structures. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(4):557–579, 2011.

[38] Noel M OâĂŹBoyle. Towards a universal smiles representation-a standard method to generate canonical smiles based on the inchi. *Journal of cheminformatics*, 4(1):22, 2012.

[39] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[40] Daylight Toolkit. Daylight chemical information systems. *Inc.: Aliso Viejo, CA*, 2007.

[41] Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. Inchi-the worldwide chemical structure identifier standard. *Journal of cheminformatics*, 5(1):7, 2013.

[42] Hamza Hentabli, Faisal Saeed, Ammar Abdo, and Naomie Salim. A new graph-based molecular descriptor using the canonical representation of the molecule. *The Scientific World Journal*, 2014, 2014.

[43] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

[44] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[45] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[46] B Chen and AJ Butte. Leveraging big data to transform target selection and drug discovery. *Clinical Pharmacology & Therapeutics*, 99(3):285–297, 2016.

[47] John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[49] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.